

Date=13/08/2020

Lecture By=Shubham Joshi

Subject ⇒ Questions regarding Trees

IN PREVIOUS LECTURE (QUICK RECAP) Date-12/08/2020	In Today's Lecture (Overview)
Types of Traversal in Tree ⇒ Preorder Traversal In Tree ⇒ InOrder Traversal ⇒ Postorder Traversal Question That Was Solved/Discussed Regarding Traversal MCQs Questions For self practice/Assignment for the Day	Question=1 Given A Binary tree Find Leaf Nodes In It Question=2 Write A program to find the height of the binary tree Question 3 Given a binary tree, return all root-to-leaf paths. MCQs Questions for self practice // CC For the day

Question=1 Given A Binary tree Find Leaf Nodes In It

== Given a binary tree, the task is to print all the leaf nodes of the binary tree from right to left

Examples:

```
Input :
      1
     / \
    2   3
   / \ / \
  4  5 6  7
Output : 7 6 5 4
```

```
Input :
      1
     / \
    2   3
   / \   \
  4  5   6
     / \ / \
    7  8 9
Output : 9 8 7 4
```

Recursive Approach: Traverse the tree in Preorder fashion, by first processing the root, then right subtree and then left subtree and do the following:

- >Check if the root is null then return from the function.
- >If it is a leaf node then print it.
- >If not then check if it has right child, if yes then call function for right child of the node recursively.
- >Check if it has left child, if yes then call function for left child of the node recursively.

Below is the implementation of the above approach:

```
# Python3 program to print
# leaf nodes from right to left

# Binary tree node
class newNode:

    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Function to print leaf
# nodes from right to left
def printLeafNodes(root):

    # If node is null, return
    if root == None:
        return

    # If node is leaf node,
    # print its data
    if (root.left == None and
        root.right == None):
        print(root.data, end = " ")
        return

    # If right child exists,
    # check for leaf recursively
    if root.right:
```

```

        printLeafNodes(root.right)

    # If left child exists,
    # check for leaf recursively
    if root.left:
        printLeafNodes(root.left)

# Driver code
root = newNode(1)
root.left = newNode(2)
root.right = newNode(3)
root.left.left = newNode(4)
root.left.right = newNode(5)
root.right.left = newNode(6)
root.right.right = newNode(7)
root.left.left.left = newNode(8)
root.right.right.left = newNode(9)
root.left.left.left.right = newNode(10)

printLeafNodes(root)

```

Output

```

E:\Study\Codes\if>C:/python/pytho
9 6 5 10
E:\Study\Codes\if>

```

Question Link

<https://www.geeksforgeeks.org/print-all-leaf-nodes-of-a-binary-tree-from-right-to-left/>

Question=2Write A program to find the height of the binary tree

Explanation

Given a binary tree, find the height of it. Height of the empty tree is 0 and the height of the tree below is 3.

Recursively calculate height of left and right subtrees of a node and assign height to the node as max of the heights of two children plus 1. See below pseudo code and program for details.

Algorithm:

maxDepth()

1. If tree is empty then return 0

2. Else

(a) Get the max depth of left subtree recursively i.e.,
call maxDepth(tree->left-subtree)

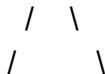
(a) Get the max depth of right subtree recursively i.e.,
call maxDepth(tree->right-subtree)

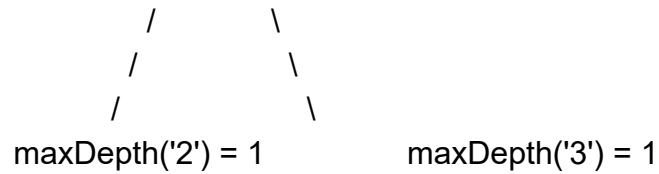
(c) Get the max of max depths of left and right
subtrees and add 1 to it for the current node.

max_depth = max(max dept of left subtree,
max depth of right subtree)
+ 1

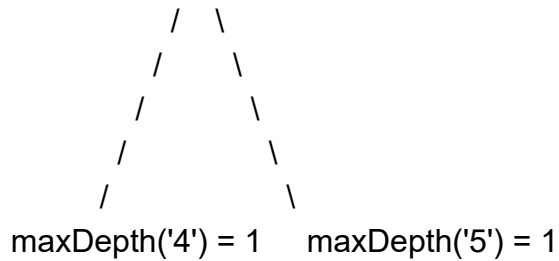
(d) Return max_depth

See the below diagram for more clarity about execution of the recursive function maxDepth() for above example tree.

$$\begin{aligned} \text{maxDepth('1')} &= \text{max}(\text{maxDepth('2')}, \text{maxDepth('3')}) + 1 \\ &= 2 + 1 \end{aligned}$$




= max(maxDepth('4'), maxDepth('5')) + 1
 = 1 + 1 = 2



Implementation/Code:

```
# Python3 program to find the maximum depth of tree

# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Compute the "maxDepth" of a tree -- the number of nodes
# along the longest path from the root node down to the
# farthest leaf node
def maxDepth(node):
    if node is None:
        return 0 ;

    else :

        # Compute the depth of each subtree
        lDepth = maxDepth(node.left)
```

```

        rDepth = maxDepth(node.right)

        # Use the larger one
        if (lDepth > rDepth):
            return lDepth+1
        else:
            return rDepth+1

# Driver program to test above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print ("Height of tree is %d" %(maxDepth(root)))

```

Output

```

E:\Study\Codes\if>C:/pyt
Height of tree is 3

```

Question Link

<https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/>

Question 3 Given a binary tree, return all root-to-leaf paths.

Note: A leaf is a node with no children.

Example:

Input:

```

      1
     / \
    2   3

```

\
5

Output: ["1->2->5", "1->3"]

Explanation: All root-to-leaf paths are: 1->2->5, 1->3

Code

```
class Solution:
    def binaryTreePaths(self, root):
        res = []
        def dfs(n, A):
            if n:
                A.append( str(n.val) )
                if (not n.left) and (not n.right):
                    res.append( '->'.join(A) )
                else:
                    dfs(n.left ,A)
                    dfs(n.right,A)
                A.pop()
        dfs(root, [])
        return res
```

Question link

<https://leetcode.com/problems/binary-tree-paths/submissions/>

MCQs

1.What is the time complexity for finding the lower bound of an element in an array ?

A= $O(N)$

B= $O(\log n)$

C= $O(1)$

2.What is the worst height of a tree

$$A=n^2$$

$$B=n$$

$$C=\log n$$

3. what is the height of a complete tree or full binary tree?

$$A=\log n$$

$$B=n$$

$$C=1$$

4. what is the complexity for finding the height of the binary tree?

$$A=O(n)$$

$$B=O(\log n)$$

$$C=O(n^2)$$

Questions for self practice // CC For the day

1. <https://practice.geeksforgeeks.org/problems/count-leaves-in-binary-tree/1>
2. <https://practice.geeksforgeeks.org/problems/height-of-binary-tree/1>
3. <https://leetcode.com/problems/binary-tree-paths/submissions/>